AFBR-S50 series
Time-of-Flight sensor modules for distance and motion measurement

Release Notes
Argus Productive SDK
Release V1.3.5
August 12th, 2021

The SDK Release V1.3.5 is a Maintenance Software Release, which shall be used together with AFBR-S50 TOF modules. All Hardware revisions to date (both V1.1 as well as V1.2 devices) are supported.

The Software supports the following AFBR-S50 variants:

AFBR-S50MV85G (medium range, typical up to 10 m, 7… 16 Pixels illuminated), productive

AFBR-S50MV85I (short to medium range, typical up to 5 m, 32 Pixels illuminated), productive

AFBR-S50MV68B (medium range, typical up to 10 m, 1 … 2 Pixel illuminated), productive

AFBR-S50LV85D (long range, typical up to 30 m, 1 … 3 Pixel illuminated), productive

The Software additionally has preliminary support for the following AFBR-S50 variant:

AFBR-S50LX85D (extra-long range, typical up to 50 m, 1 … 3 Pixel illuminated), engineering

Important remark:

With the basic SDK version, no direct register access is possible and any violations of laser class 1 eye safety limits are blocked.

1. Feature enhancements from previous release 1.2.3

Id	Feature	Description	Application Note
1	Regulation of Laser Bias and average power over temperature	The Software now includes adaption of both laser bias and integration depth over temperature.	Improves stability of the sensor performance over temperature, especially for precision.
2	Compensation of crosstalk drift over temperature	The Software includes a model “predictor”, which calculates and compensates the drift of electrical crosstalk over temperature.	The color error as well as absolute error over temperature is improved, especially for dark targets.
3	Compensation of offset drift over temperature	Offset drift, which is not caused by laser bias drift, is additionally compensated	Improves absolute error deviation over temperature for all target materials.
4	Signal Quality Indicator	A signal Quality Indicator is provided, which is indicating the reliability of the signal after pixel binning. The status 107 (signal underflow) has been deprecated.	Signal Quality indicates 100% for good signals, 50% and lower for weak signals (similar to former status 107). Signal quality of 1% is an errored signal (not reliable). Signal Quality of 0% is unknown.
5	HAL Self-Test	Separate source files with test cases that verify the correct implementation of HAL on a new target platform. SPI, EEPROM and timer layers are tested.	Helps to improve confidence when porting to another platform and reduces frequent errors.
6	Direct Support of STM32 Cortex-M4	Especially for 3kHz operation, an alternative Evaluation Platform based on ARM Cortex-M4 is directly supported with binaries	In addition to that a detailed step by step Porting Guide is available, which shows how to port the code to different Hardware Platforms.
7	Additional option to select average instead of maximum binned Amplitude for Dynamic Configuration Adaption (DCA).	By default SW is configured to avoid overloaded pixels as far as possible by always reacting on the “strongest” pixel. Due to reduction of global integration depth in presence of large signals on a small number of pixels, sensitivity of pixels with weaker signals is reduced.	In case average amplitude is selected, amplitude-based DCA parameters might need to be changed. The reaction of DCA on saturated pixels as well as amplitude thresholds for regulation (max. / target / min. amplitude) can be configured with API.

2. Fixes for issues of previous release 1.2.3

Id	Issue	Description	Fix
1	Measurement errors at short distances and bright objects	The previous Software reported too large ranges in case of a bright object and short distances.	Adapt compensation algorithms at short distances.
2	Over- and undershoots of distance range after sudden strong remission changes	After a sudden strong remission change (e.g. retro reflector enters the Field of View), Distance values can differ from actual value for a few frames	Over- and undershoots as well as switching behavior have been reduced.
3	Sporadic reporting of valid ranges in outdoor “open sky” setting	If the sensor points to the sky and should report “infinity”, shotnoise can lead to sporadic wrong “valid” measurements.	The status 108 now reports “no object” reliably in case of too low Amplitudes. Error -110 (distance stalled) is only reported for errored frames with valid Amplitudes.
4	Laser Safety Alarm robustness.	Laser Amplitude checks could be influenced by high electrical crosstalk or ambient light and lead to false or delayed alarms In case of a Hardware failure	Implemented separate checks and thresholds for Reference Pixel and active Pixels as well as threshold scaling with integration time.
5	Laser timeout check (compare actual duration of a measurement with the expected time).	Laser Safety Timeout was persistent and not cleared once the error was removed. Sporadic Timing issues could have lead to a stalled system.	Clear Laser safety alarm in case the timing issue is resolved.
6	Sporadic timing issues with SPI frequencies > 15MHz	The software sporadically reduced internal frequency by 50%, which could limit the max. SPI frequency to 15MHz.	Bug removed in PLL lock polling process.

3. Known Issues and limitations

Id	Issue	Description	Recommendation	Fix planned
1	Reduced pixelcount for framerates >1kHz	Currently a reduction of pixelcount is not supported by API	Low level configurations are available, please contact your local Broadcom support	Upcoming Software Release
2	Multiple Sensor support per Microcontroller with parallel integration and evaluation	Currently multiple sensors per Microcontroller are only supported with separate instances in a sequential mode.	Use separate instances for each module, trigger measurement and evaluate only one sensor at a time.	Upcoming Software Release
3	Memory Footprint	Due to added features, the memory footprint for the argus_hnd_t data structure has been increased from 4kByte to 4.2kByte.	Increase static RAM size allocated for each sensor from 4kByte to 4.2kByte.	Upcoming Software Release
4	Remaining short distance errors	For distances below 15cm the distance error increases with target remission	In case the application is focusing on short distances only, using fixed low APD gain and low optical power is recommended and at the same time increasing the minimum analog integration depth can help in addition.	Future Software Release

4. API interface changes compared to previous release 1.2.3

i) Changed Range Offset value formats from Q9.22 to Q1.15

In order to optimize memory footprint and create a uniform data type for all range offset values, the global range offset has been changed from Q9.22 to Q1.15 data format.

In File “include/api/argus_api.h”:

“argus_api.h” v1.2.3	“argus_api.h” v1.3.5
status_t Argus_SetCalibrationGlobalRangeOffset(argus_hnd_t * hnd, argus_mode_t mode, q9_22_t value);	status_t Argus_SetCalibrationGlobalRangeOffset(argus_hnd_t * hnd, argus_mode_t mode, q0_15_t value);
status_t Argus_GetCalibrationGlobalRangeOffset(argus_hnd_t * hnd, argus_mode_t mode, q9_22_t * value);	status_t Argus_GetCalibrationGlobalRangeOffset(argus_hnd_t * hnd, argus_mode_t mode, q0_15_t * value);

How to Migrate:

Convert from old q9_22_t type to q0_15_t type and back via:

```
q0_15_t value_0_15 = …;
q9_22_t value_9_22 = value_0_15 << 7;
q0_15_t value_0_15 = (q0_15_t)(value_9_22 >> 7);
```

Note that the range and granularity of Q0.15 data format is less than Q9.22 format!

ii) Replaced SampleCount by SampleTime parameter for range offset calibration and crosstalk calibration configuration:

In order to remove the influence of the measurement framerate on the duration of a calibration sequence, the sample count parameter has been changed to a sample time parameter. Note that the value was number of frames in v1.2.3 and is a time parameter in milliseconds in v1.3.5.

In File “include/api/argus_api.h”

“argus_api.h” v1.2.3	“argus_api.h” v1.3.5
status_t Argus_SetCalibrationRangeOffsetSequenceSampleCount(argus_hnd_t * hnd, uint16_t value);	status_t Argus_SetCalibrationRangeOffsetSequenceSampleTime(argus_hnd_t * hnd, uint16_t value);
status_t Argus_GetCalibrationRangeOffsetSequenceSampleCount(argus_hnd_t * hnd, uint16_t * value);	status_t Argus_GetCalibrationRangeOffsetSequenceSampleTime(argus_hnd_t * hnd, uint16_t * value);
status_t Argus_SetCalibrationCrosstalkSequenceSampleCount(argus_hnd_t * hnd, uint16_t * value);	status_t Argus_SetCalibrationCrosstalkSequenceSampleTime(argus_hnd_t * hnd, uint16_t value);
status_t Argus_GetCalibrationCrosstalkSequenceSampleCount(argus_hnd_t * hnd, uint16_t * value);	status_t Argus_GetCalibrationCrosstalkSequenceSampleTime(argus_hnd_t * hnd, uint16_t * value);

How to Migrate:

Replace the corresponding function and convert from sample count to sample time via frame time parameter:

```
uint16_t sample_time_ms = sample_count * frame_time_ms;
uint16_t sample_count = sample_time_ms / frame_time_ms;
```

iii) Changed DCA Laser Current Setting to DCA Power Stage Selector

Since all calibration is done at a specified laser modulation current, it is not possible to change the laser output current. Instead, a power stage selector has been implemented that allows to select between HIGH, LOW or AUTO stages. The MEDIUM_HIGH and MEDIUM_LOW stages have been dismissed.

In File “include/api/argus_dca.h”

“argus_dca.h” v1.2.3	“argus_dca.h” v1.3.5
typedef enum { DCA_POWER_LOW = 0, DCA_POWER_MEDIUM_LOW = 1, DCA_POWER_MEDIUM_HIGH = 2, DCA_POWER_HIGH = 3 } argus_dca_power_t;	typedef enum { DCA_POWER_LOW = 0, DCA_POWER_HIGH = 1, DCA_POWER_AUTO = 2 } argus_dca_power_t;
typedef struct { // … uq12_4_t PowerNom; uq12_4_t PowerMin; // … } argus_cfg_dca_t;	typedef struct { // … argus_dca_power_t Power; // … } argus_cfg_dca_t;

How to Migrate:

Use the default value “DCA_POWER_AUTO”. If limitation of output power is really required, use “DCA_POWER_LOW” instead.

iv) The Measurement Status 107: UNDERFLOW has been dismissed

The status that was set when the DCA reached its lowest settings (107: UNDERFLOW) has been removed. The state where the DCA is in its lowest values is a common use case for dark or far targets and does not indicate a degraded measurement signal quality.

In File “include/api/argus_status.h”

The following line is removed:

```
STATUS_ARGUS_UNDERFLOW = 107,
```

How to Migrate:

In case a hint on weak but still valid signal is required, use the newly implemented “WEAK_SIGNAL” flag instead:

```
argus_results_t * res;

// after EvaluateData …

if (res->Frame.State & ARGUS_STATE_WEAK_SIGNAL)
    // …
```